Car purchase prediction system

BY

AVRIT M

ROHIT BHARADWAJ

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

5.2 Solution Architecture

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

6.2 Sprint Planning & Estimation

6.3 Sprint Delivery Schedule

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. PERFORMANCE TESTING

8.1 Performance Metrics

9. RESULTS

9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1.Introduction

1.1 Overview

Car Purchase prediction system is a machine learning trained model which can be used to predict whether an individual is likely to purchase a car in the near future.For consumers, purchasing a car is a significant decision that involves a substantial financial commitment. It is essential to make an informed choice that aligns with one's preferences, needs, and budget. However, the abundance of choices can make this process overwhelming. This is where predictive analytics and machine learning can play a pivotal role. The Car Purchase Prediction Project aims to simplify the car buying process by leveraging data-driven insights to assist potential buyers in making informed decisions. By analyzing historical data on various car models, customer preferences, and market trends, this project seeks to predict the most suitable car options for individual customers, enhancing their overall buying experience

Project Objectives:

Identify key factors that influence car purchase decisions

Collect and prepare relevant data for model training

Train and evaluate various machine learning algorithms

Deploy the best-performing model into a user-friendly application

Model Deployment and Evaluation:

Integrate the best-performing model into a user-friendly application or web service

Continuously monitor model performance and retrain as needed

Challenges and Considerations:

Data quality and availability

Feature selection and data preprocessing

Model interpretability and explain ability

Expected Outcomes:

Improved marketing ROI through targeted car purchase predictions

Reduced risk of loan defaults and insurance fraud

1.2 Purpose

The purpose of this project is to develop a car purchase prediction system that can accurately identify individuals who are likely to purchase a car in the near future. This system will be used to improve marketing ROI, reduce risk of loan defaults and insurance fraud, optimize inventory levels and supply chain management, and enhance customer satisfaction.

2.LITERATURE SURVEY

2.1 Existing problem

Buyers visit multiple dealerships to compare prices and features, and the negotiation process can be complex and stressful. They do not know if a car can be purchased or not due to their budget and income.

This in turn affects the car manufacturers in the marketplace as they do not know if customer will make a purchase or not. Such cases can lead to wastage of resources in the automobile production and can reduce the value of the company in the market.

2.2 References

A review of the current automotive manufacturing practice from an energy perspective by *A. Giampieri , J. Ling-Chin*

Assessment of strategic raw materials in the automobile sector *by Abel Ortego , Guiomar Calvo*

2.3 Problem Statement Definition

Problem: Automobile manufacturers and dealerships need to be able to predict which customers are likely to purchase a car in the near future. This information can be used to target marketing campaigns and improve inventory management.

Requirement: Develop a car purchase prediction system that can accurately predict which customers are likely to purchase a car in the next 6-12 months. The system should be able to use a variety of data sources, such as customer demographics, purchase history, and browsing behavior, to make its predictions.

Benefits:

Improved marketing campaigns: Automobile manufacturers and dealerships can use the predictions from the car purchase prediction system to target their marketing campaigns more effectively. This can lead to increased brand awareness and sales.

Better inventory management: Automobile manufacturers and dealerships can use the predictions from the car purchase prediction system to better manage their inventory. This can help to ensure that they have the right cars in the right place at the right time.

Reduced waste: By accurately predicting which customers are likely to purchase a car, automobile manufacturers and dealerships can reduce waste. This can be done by avoiding overproduction and by ensuring that they have the right cars in stock.

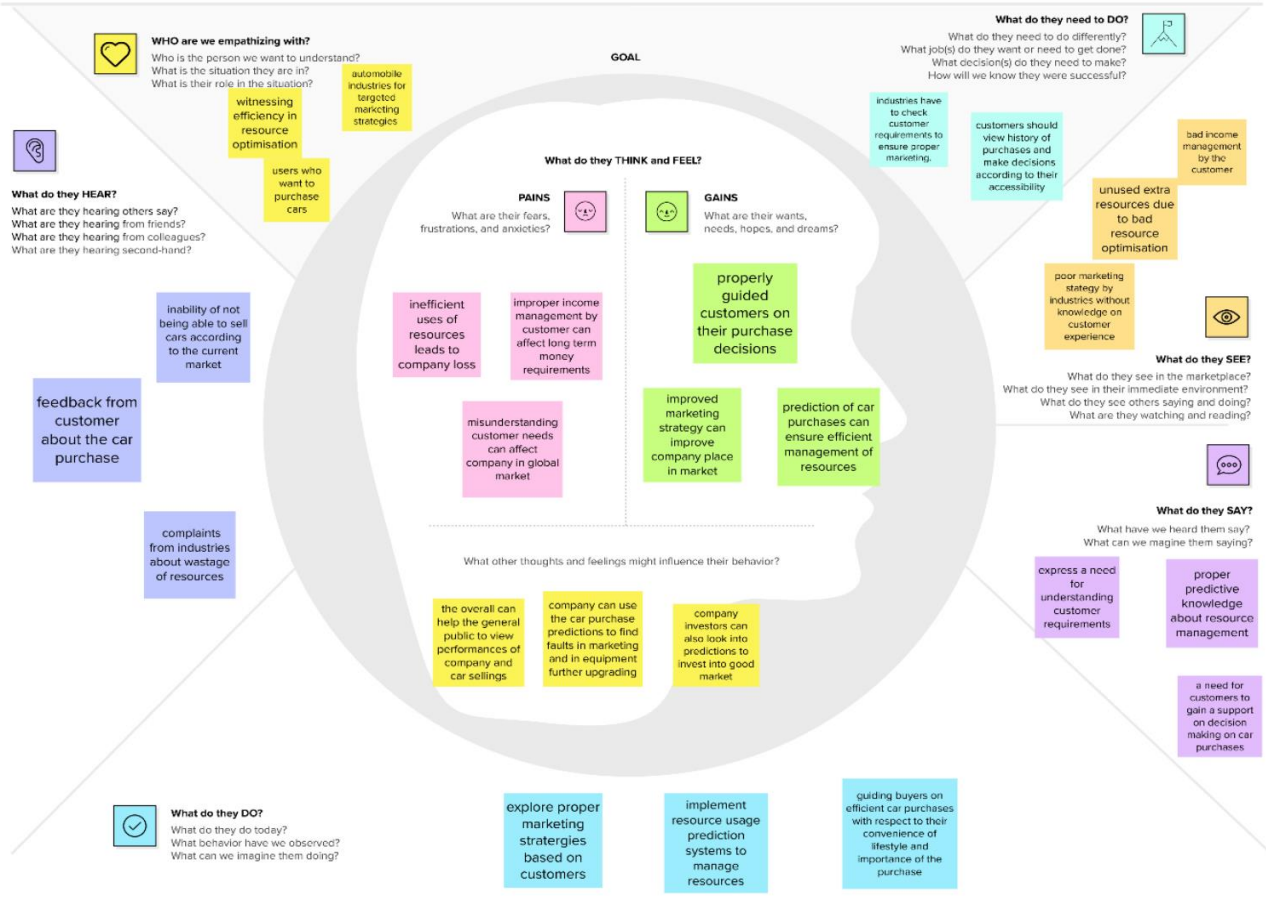
Constraints:

The car purchase prediction system should be able to make accurate predictions with a high degree of confidence.

The car purchase prediction system should be able to be implemented in a cost-effective manner.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



The objective of this discovery is to empathize with the customer and their issues, thoughts and decisions regarding the requirement of car purchase prediction in their life. We list out the needs of the buyer and note their opinions on what is the existing issue, what the hear, see and say regarding it, and empathize, which helps us to arrive at a conclusion to develop our model.

3.2 Ideation & Brainstorming

BRAINSTORMING IDEAS AND VOTING

Brainstorming ideas is a creative process where a group generates a listof potential solutions, suggestions, or concepts for a specific problem or project. Voting in brainstorming involves

participants selecting and prioritizing their favourite or most promising ideas from the list to determine which ones should be pursued further.

BRAINSTORMING FOR CAR PURCHASEPREDICTION

The objective of this brainstorming session is to generate creative and practical ideas to address the issue of car purchase prediction .We aim to develop a model assists potential buyers by estimating their likelihoodto make a purchase, guiding decision-making.

The brainstorming session will include a diverse group of stakeholders, including car professionals, buyers, sellers and technology enthusiasts.The diversity will ensure a wide range of perspective and ideas

Step-1: Team Gathering, Collaboration and Select theProblem Statement

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

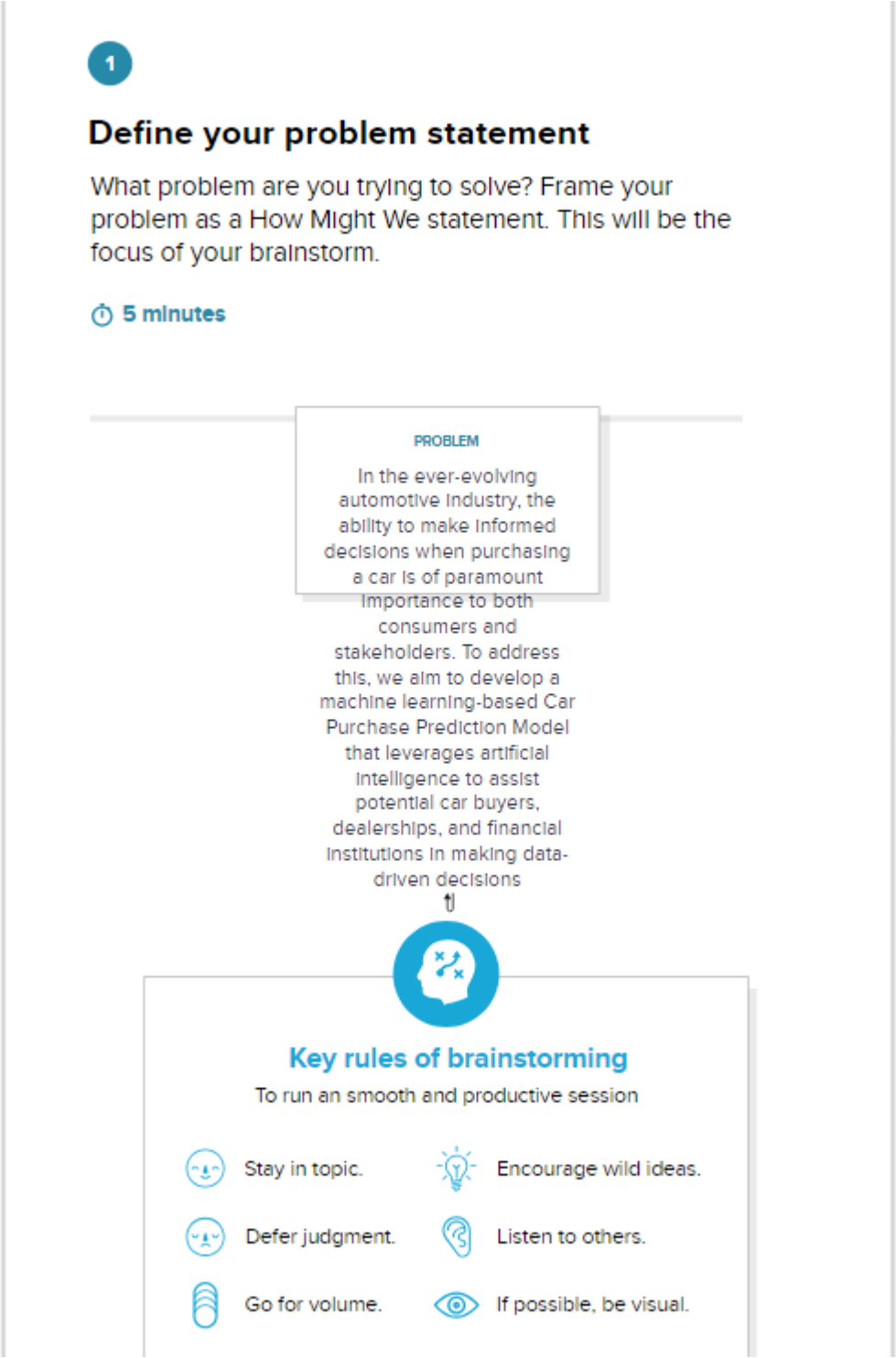
10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →



Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

AVRIT

Analyze customer reviews and sentiment to understand what people like and dislike about different car models.

Develop a model using AI and machine learning algos that predicts the market price of a car based on features like make, model, year, mileage, and location.

Predict the fuel efficiency of a car model based on its specifications, allowing consumers to make more informed choices.

Predict the safety rating of a car based on crash test results, safety features, and manufacturer history.

Build an application that helps users customize their dream car based on their preferences and budget.

ROHIT

Analyze trends in car purchases by region, make, and model. This can help car manufacturers and dealerships understand market demands and adapt their strategies accordingly.

Develop a model that estimates the current value of a used car based on its age, mileage, condition, and market trends.

Build a recommendation system that suggests cars to potential buyers based on their preferences, budget, and past behavior.

Predict the future resale value of a car based on its current state and depreciation trends.

Develop a tool that generates buyer personas for car models, helping marketing teams target their campaigns effectively.

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

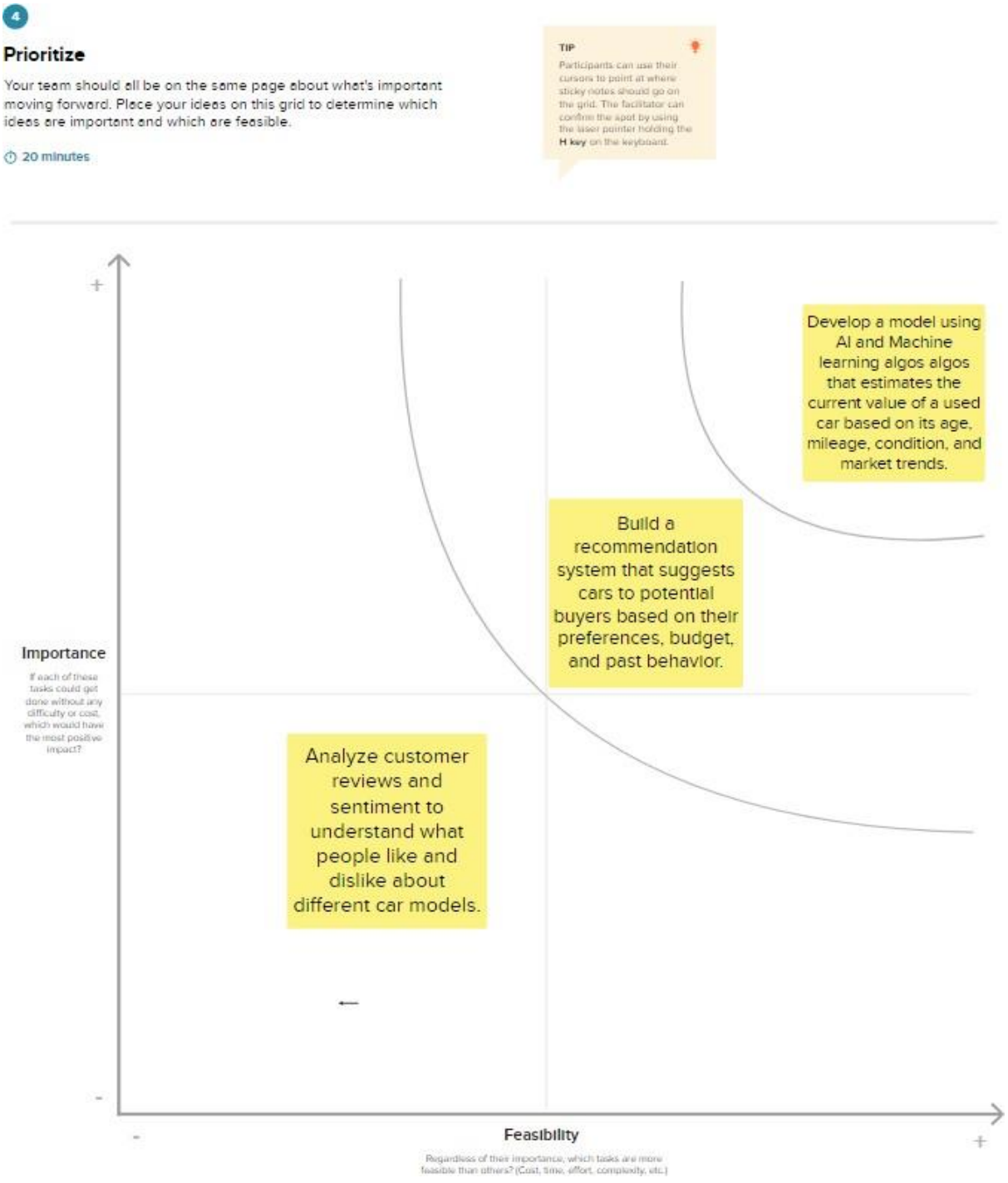
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Develop a model using AI and machine learning algos algosthat estimates the current value of a used car based on its age, mileage, condition, and market trends.

Build a recommendation system that suggests cars to potential buyers based on their preferences, budget, and past behavior.

Analyze customer reviews and sentiment to understand what people like and dislike about different car models.

Step-3: Idea Prioritization



Certainly, here's a description for why the idea of "CAR PURCHASE PREDICTION using AI and ML

algorithms " was chosen over the other options:

We have decided to prioritize the implementation of CAR PURCHASEPREDICTION using AI and ML algorithms for several compelling reasons.

Firstly, this idea offers the potential for a substantial and immediateimpact on our car purchase predictions . By leveraging artificial intelligence and ML algos such as regressor,classifiers we can significantly improve the efficiency and accuracy of car purchase predictions sorting.

Additionally, our commitment to customers is well-aligned with this choice. AI amd ML algos makes this project feasible and can acheievhigher accuracy rates and good performance metrics .

In conclusion, the selection of CAR PURCHASE PREDICTION using AIand ML algorithms as our top priority is a strategic decision based on itshigh impact potential, feasibility, and alignment with our goals.

MURAL LINK:

https://app.mural.co/t/rohit5856/m/rohit5856/1697539105901/4ce29ef1776a09d1a4234eff73918fa068cb68a7?sender=ua911e88567ca9674d1a64829

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

The car purchase prediction system should be able to accept data from a variety of sources, such as customer demographics, purchase history, and browsing behavior.

The car purchase prediction system should be able to use this data to train a machine learning model to predict which customers are likely to purchase a car in the next 6-12 months.

The car purchase prediction system should be able to generate predictions for individual customers or for groups of customers.

The car purchase prediction system should be able to provide a confidence score for each prediction.

The car purchase prediction system should be able to be integrated with other systems, such as CRM systems and marketing automation systems.

4.2 Non-Functional requirements

Accuracy: The car purchase prediction system should be able to achieve an accuracy of at least 80%.

Scalability: The car purchase prediction system should be able to handle a large volume of data and users.

Performance: The car purchase prediction system should be able to generate predictions in a timely manner.

Reliability: The car purchase prediction system should be reliable and stable.

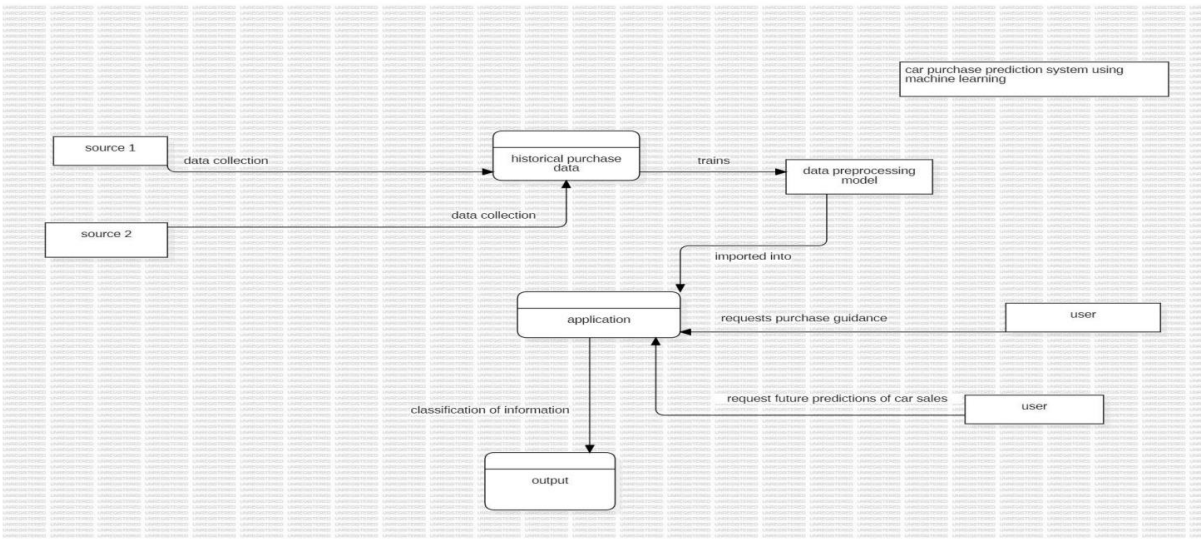
Security: The car purchase prediction system should be secure and protect customer data.

Usability: The car purchase prediction system should be easy to use and understand.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

Car Purchase Prediction System DFD



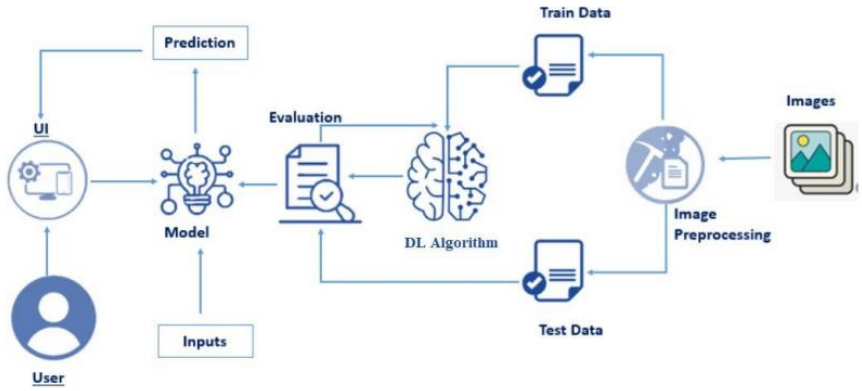
Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically.

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Car companies	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the car purchase prediction project.	successfully configured with all necessary tools and frameworks	High	Sprint 1
Data Engineers	development environment	USN-2	Gather a diverse dataset of informations containing different types of data(model ,price ,mileage etc) for training the machine learning model.	Gathered a diverse dataset of information depicting various data on cars	High	Sprint 1
Data Engineers	Data collection	USN-3	Preprocess the collected dataset by encoding catagorical data,feature scaling values, and splitting it into training and validation sets.	preprocessed the dataset	High	Sprint 2
Researchers and Academics	data preprocessing	USN-4	Explore and evaluate different machine learning architectures (e.g., RandomForest, NaiveBayes, KNN etc) to select the most suitable model for car purchase prediction	we could explore various ML models	High	Sprint 2
Data scientist/Analysts	model development	USN-5	train the selected machine learning model using the preprocessed dataset and monitor its performance on the validation set.	we could do validation	High	Sprint 3
Data scientist/Analysts	Training	USN-6	implement regularization techniques and avoiding overfitting,underfitting to improve the model's robustness and accuracy.	we could do testing	medium	Sprint 3
Backend team	model deployment & Integration	USN-7	deploy the trained ml model to make it accessible for car purchase detection.	we could check the scalability	medium	Sprint 4
Backend team	Testing & quality assurance	USN-8	conduct thorough testing of the model toidentify and report any issues or bugs. fine-tune the model hyperparameters and optimize its performance based on user feedback and testing results.	we could test the quality by user reviews	medium	Sprint 5

5.2 Solution Architecture

Solution Architecture Diagram



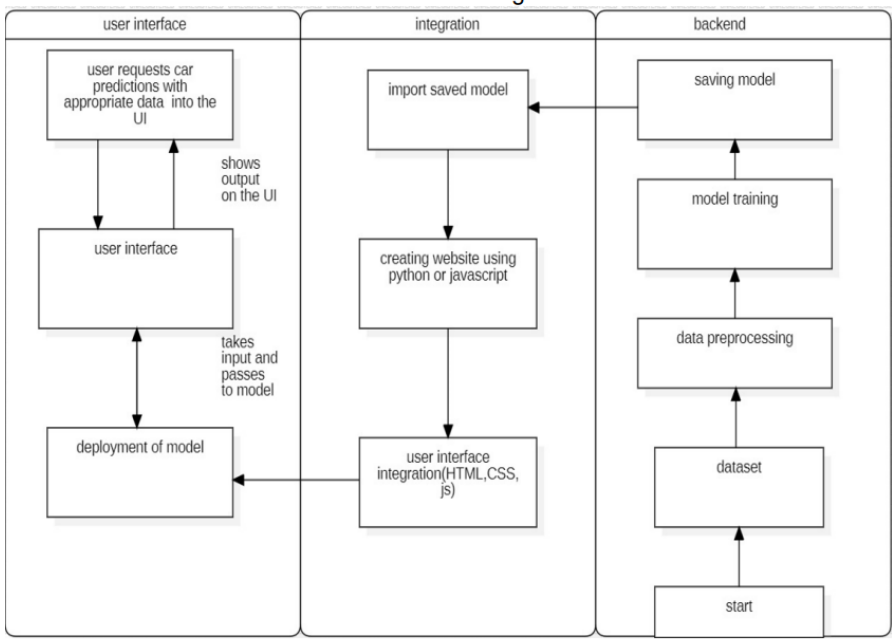
It optimizes the car purchase prediction process by leveraging various machine learning algorithms . It not only enhances accuracy but also improves the efficiency of car purchase model , contributes to accurate predictions.

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



6.2 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the Car Purchase Prediction project.	1	High	Rohit
Sprint-1	development environment	USN-2	Gather a diverse dataset of information containing different types of data(model, price) for training the machine learning model.	2	High	Rohit
Sprint-2	Data collection	USN-3	Preprocess the collected dataset by encoding categorical data and feature scaling values, and splitting it into training and validation sets.	2	High	Avrit
Sprint-2	data preprocessing	USN-4	Explore and evaluate different deep learning architectures (e.g., Random Forest,Classifiers) to select the most suitable model for Car Purchase Prediction	3	High	Avrit
Sprint-3	model development	USN-5	train the selected machine learning model using the preprocessed dataset and monitor its performance on the validation set.	4	High	Rohit
Sprint-3	Training	USN-6	implement regularization techniques and try avoiding overfitting,underfitting problems to improve the model's robustness and accuracy.	6	medium	Avrit
Sprint-4	model deployment & Integration	USN-7	deploy the trained machine learning model to make it accessible for car purchase prediction	1	medium	Avrit
Sprint-5	Testing & quality assurance	USN-8	conduct thorough testing of the model to identify and report any issues or bugs. fine-tune the model hyperparameters and optimize its performance based on user feedback and testing results.	1	medium	Rohit

6.3Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	3	3 Days	18 Oct 2023	20 Oct 2023	3	20 Oct 2023
Sprint-2	5	4 Days	21 Oct 2023	24 Oct 2023	5	24 Oct 2023
Sprint-3	10	6 Days	25 Oct 2023	30 Oct 2023		30 Oct 2023
Sprint-4	1	6 Days	31 Oct 2023	5 Nov 2023		5 Nov 2023
Sprint-5	1	4 Days	6 Nov 2023	09 Nov 2023		09 Nov 2023

7.CODING & SOLUTIONING

Car purchase predictor:

This feature allows the user to know if he/she should/would purchase a car with respect to the details entered by her.The user is required to enter his/her age,gender and salary and based upon that our trained model will predict if the purchase is probable or no

8. PERFORMANCE TESTING

1.1 Performace Metrics

After researching with multiple machine learning algorithms we came to a conclusion that knn algorithm and SVM algorithm work the best with respect to our dataset.

We have employed Knn algorithm to build the web application and integrated it using flask.
The performance metrics for Knn algorithm are

- Precision,
- Recall,
- Accuracy
- F1-Score.

They can all be viewed using classification report

```
print(classification_report(y_test,y_pred))
```

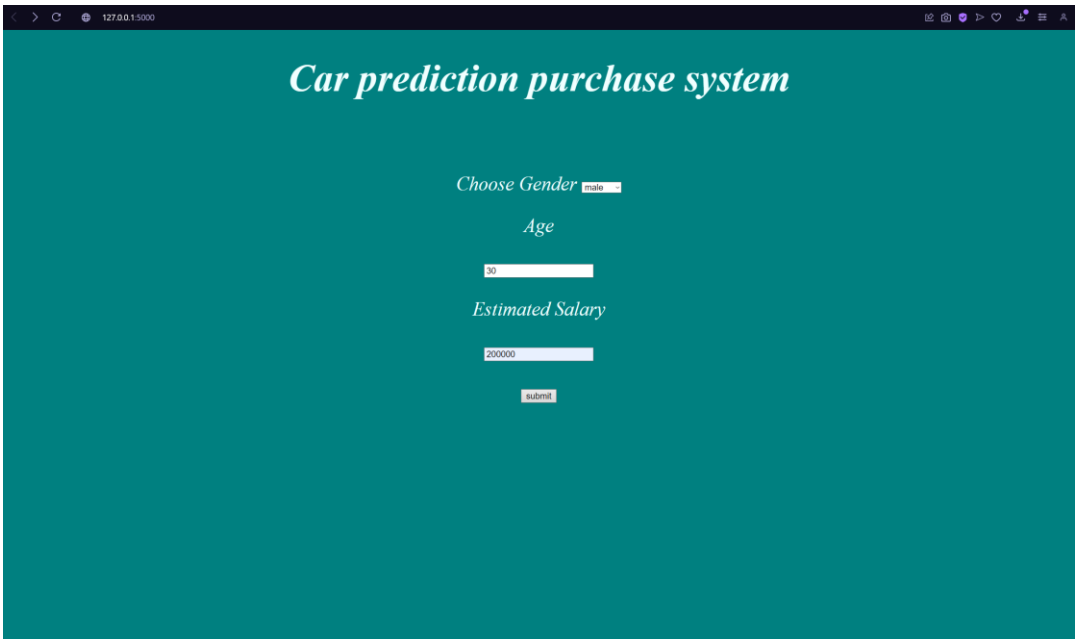
```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80
```

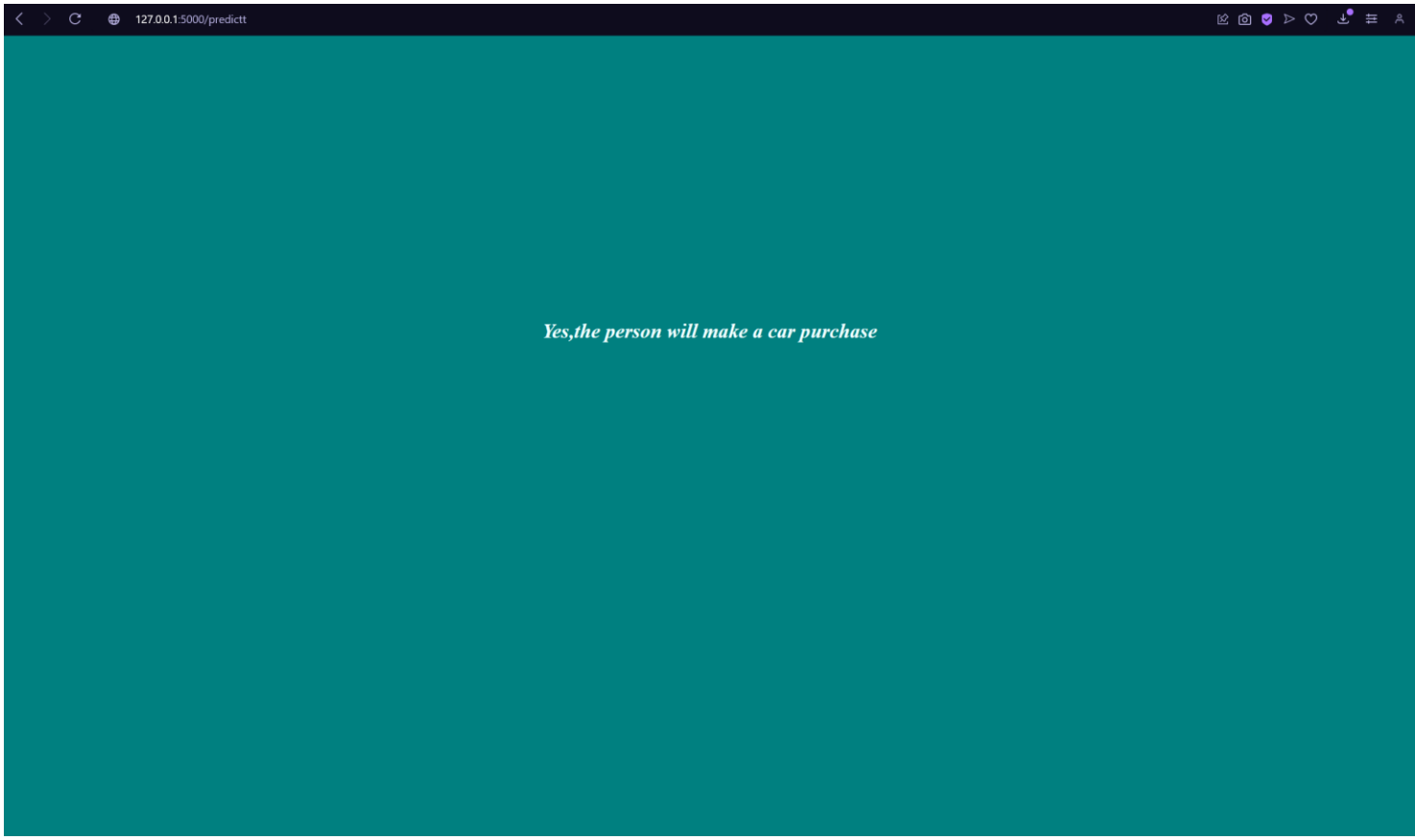
9.RESULTS

1.2 Output Screenshots

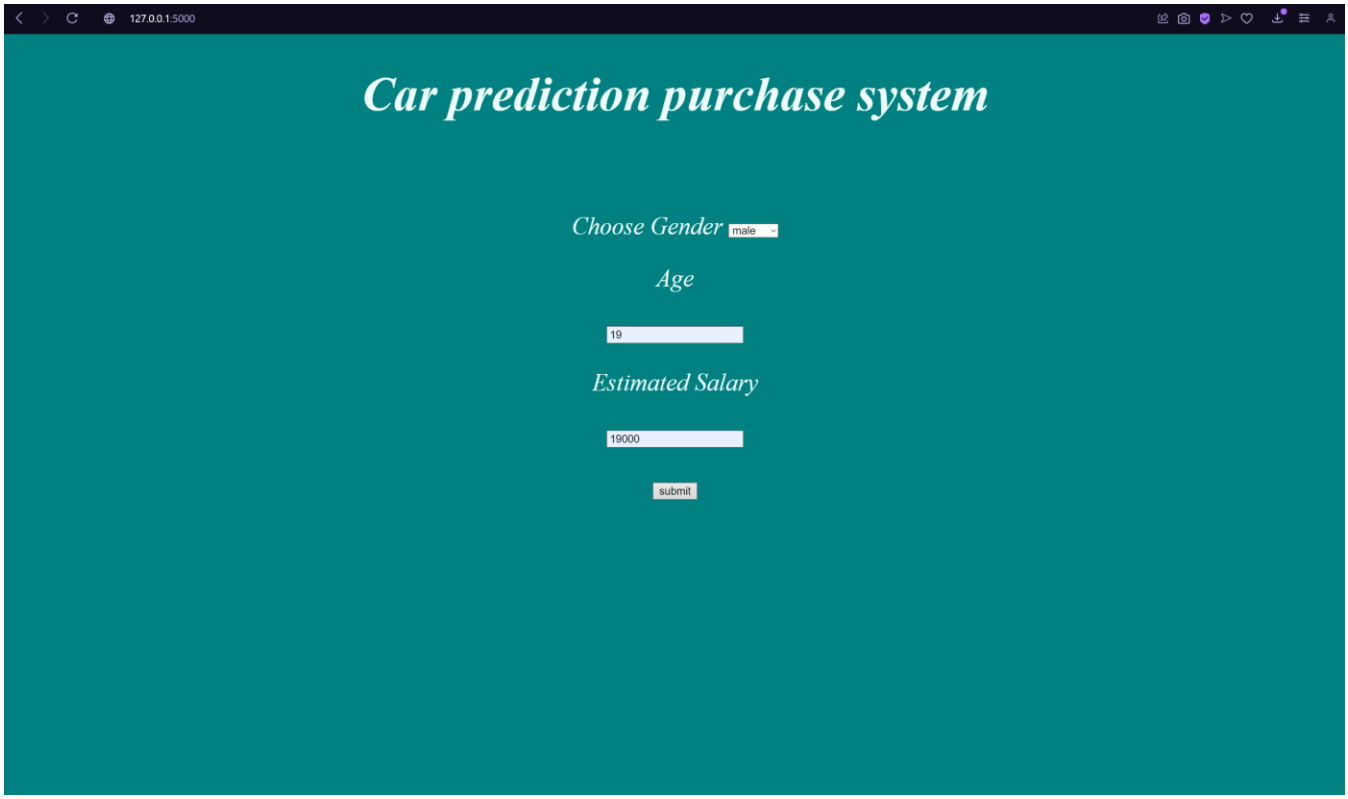
There can be atmost two predictions

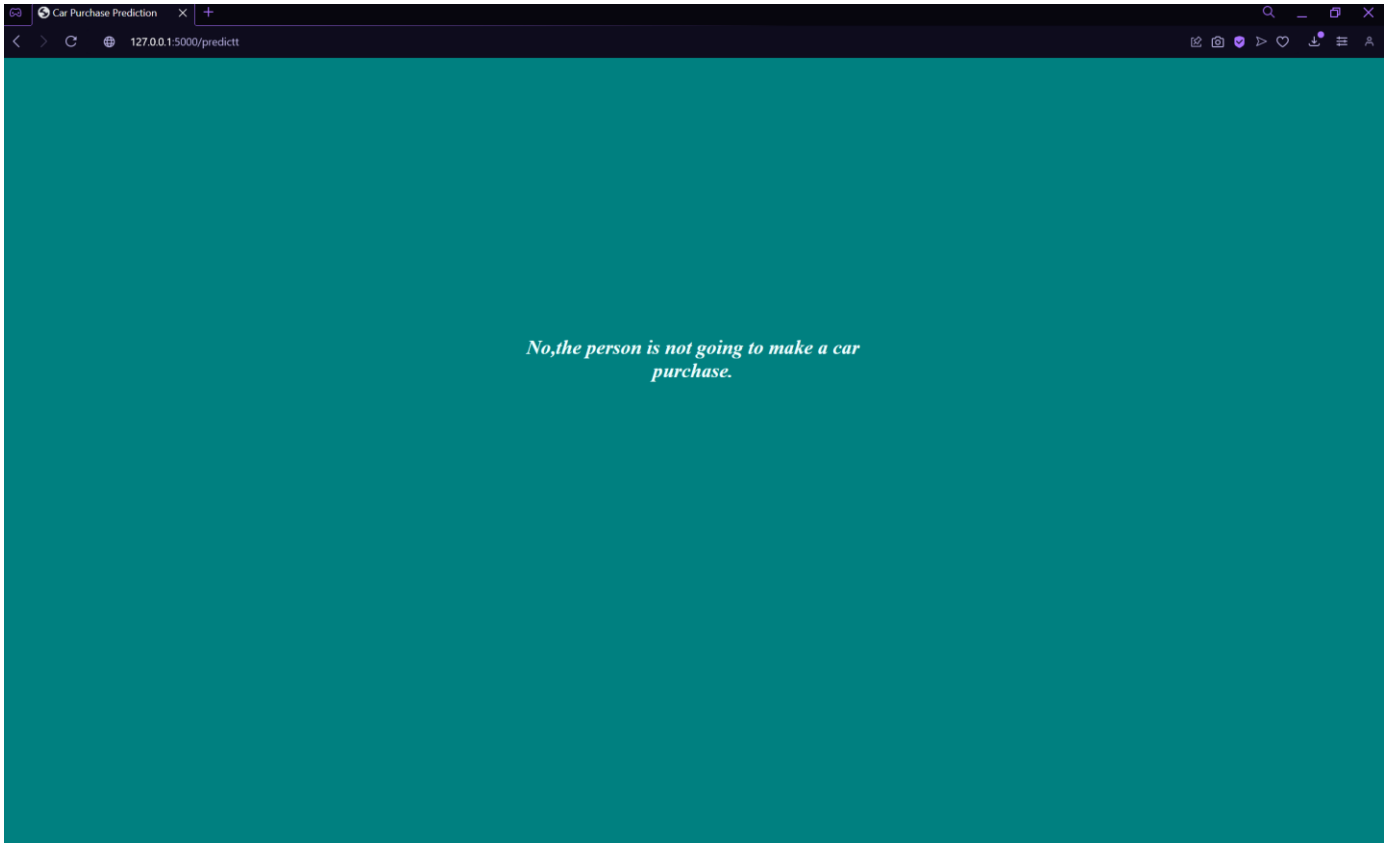
1)The purchase will be made(gender=male,age=30,salary=2,00,000)





2)The purchase wont be made(gender=male,age=19,salary=19,000)





10.ADVANTAGES & DISADVANTAGES

Advantages of a car purchase predictor: The future scope for a car purchase predictor project holds promising opportunities for enhancement and innovation. Here are some potential avenues for development:

1. **Informed Decision-Making**: It helps buyers make more informed decisions by analyzing various factors that influence car purchases, such as market trends, user preferences, and economic indicators.

2. **Financial Planning:** Buyers can plan their finances better, considering the predicted costs associated with the purchase, including depreciation, insurance, and maintenance.

3. **Time Efficiency:** It saves time for buyers by narrowing down choices based on their preferences and budget, reducing the need to visit multiple dealerships or spend excessive time on research.

4. **Customization:** Some predictors may allow users to input specific criteria, enabling a more customized prediction tailored to individual needs and preferences.

Disadvantages of a car purchase predictor:

Lack of Human Touch: Car buying can be a personal experience, and relying solely on a predictor may eliminate the human touch and expertise offered by sales professionals. The future scope for a car purchase predictor project holds promising opportunities for enhancement and innovation. Here are some potential avenues for development:

11.CONCLUSION

In conclusion, the car purchase predictor project offers a valuable tool for buyers seeking informed and efficient decision-making in the complex world of automobile purchases. By harnessing the power of data analysis and predictive algorithms, users can gain insights into market trends, financial considerations, and tailored recommendations. However, it is crucial to recognize the limitations of such predictors, acknowledging the ever-changing nature of the automotive landscape and the unique, subjective aspects of individual preferences. As technology continues to advance, integrating the benefits of a car purchase predictor with a human touch in the form of expert advice and firsthand experiences ensures a more holistic and reliable approach to navigating the exciting journey of buying a car.

12.FUTURE SCOPE

The future scope for a car purchase predictor project holds promising opportunities for enhancement and innovation. Here are some potential avenues for development:

Machine Learning Advancements: Leveraging advanced machine learning algorithms and artificial intelligence techniques can refine prediction accuracy by continuously learning and adapting to evolving market dynamics.

Integration of Real-Time Data: Incorporating real-time data feeds from various sources, including social media, automotive reviews, and economic indicators, can enhance the predictor's ability to capture current consumer sentiments and emerging trends.

Personalized User Experience: Future iterations could focus on delivering a more personalized user experience by considering individual user preferences, lifestyle factors, and previous purchase history, creating tailored recommendations for each user.

Augmented Reality (AR) Features: Introducing AR features to enable users to virtually experience a selected car model, exploring its interior, features, and even taking it for a simulated test drive, enhancing the decision-making process.

Blockchain for Transparency: Implementing blockchain technology can enhance transparency in the prediction process, ensuring that users have access to the most reliable and unaltered information, building trust in the predictions provided.

Collaboration with Dealerships: Building partnerships with dealerships to integrate the predictor directly into their platforms can streamline the buying process, allowing users to seamlessly transition from prediction to purchase within a trusted dealership network.

Environmental Impact Assessment: Including an environmental impact assessment feature that provides users with information on the eco-friendliness of a chosen vehicle, considering factors such as fuel efficiency and emissions, aligning with the growing trend of sustainable and eco-conscious consumer choices.

Continuous User Feedback Loop: Establishing a continuous feedback loop with users to gather insights on their experiences post-purchase, enabling the predictor to improve its recommendations based on real-world user feedback and satisfaction.

By embracing these future-oriented enhancements, a car purchase predictor project can evolve into a comprehensive, user-centric platform that not only guides users through the decision-making process but actively contributes to the improvement and sustainability of the automotive industry.

13.APPENDIX

Source Code

Customer Car Purchase Prediction by watching Advertisement on Social Media

Here we used Car_Purchase dataset which providing information regarding the person's age and estimated salary & if he/she is interested in buying a car .(yes=1,No=0) we will predict that what are the chances of new person of some age to be interested in buying car

IMPORTING NECESSARY LIBRARIES

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Activity 2: Import the dataset

IMPORTING DATASET

```
[ ]  # Importing the dataset
     df = pd.read_csv('Car_purchase.csv')
```

Activity 3: Check for Null values

CHECKING FOR NULL VALUES

```
df.isnull().any
```

```
<bound method NDFrame._add_numeric_operations.<locals>.any of      User ID  Gender    Age  EstimatedSalary  Purchased
0      False   False  False            False      False
1      False   False  False            False      False
2      False   False  False            False      False
3      False   False  False            False      False
4      False   False  False            False      False
..       ...     ...    ...              ...        ...
395    False   False  False            False      False
396    False   False  False            False      False
397    False   False  False            False      False
398    False   False  False            False      False
399    False   False  False            False      False

[400 rows x 5 columns]>
```

```
[ ]  df.isnull().sum()
```

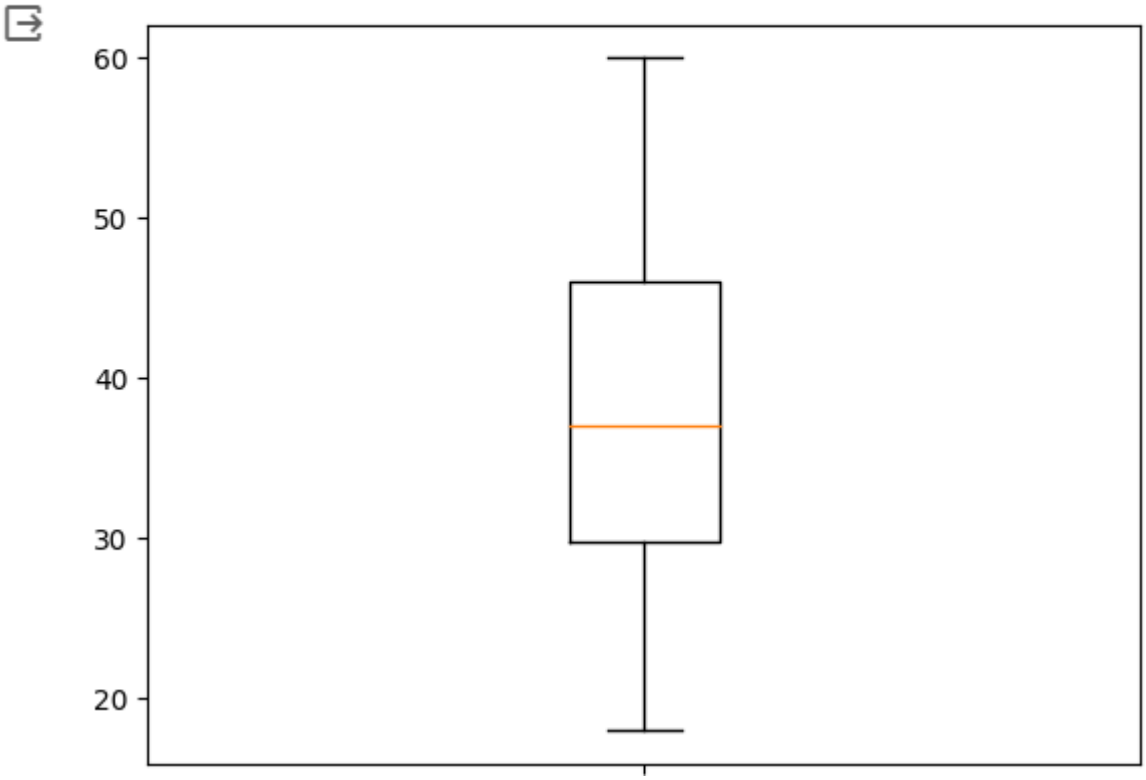
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

As we can see there are no null values

Activity 4: Data Visualization

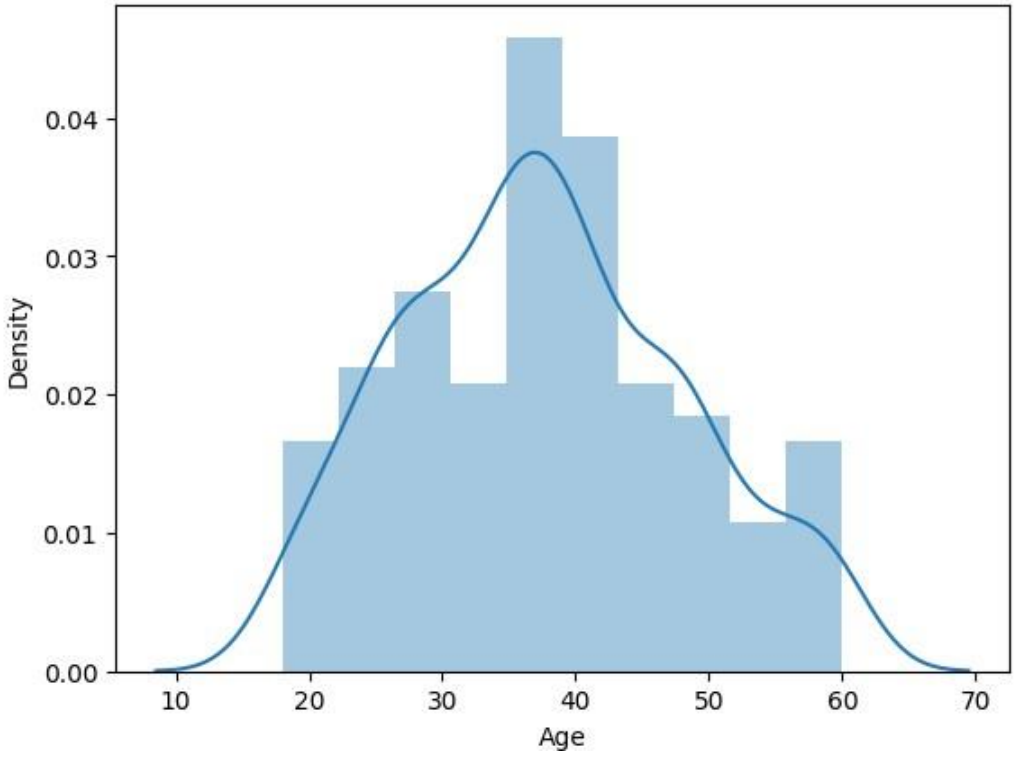
UNIVARIATE ANALYSIS

```
q = list(df.Age)
plt.boxplot(q)
plt.show()
```

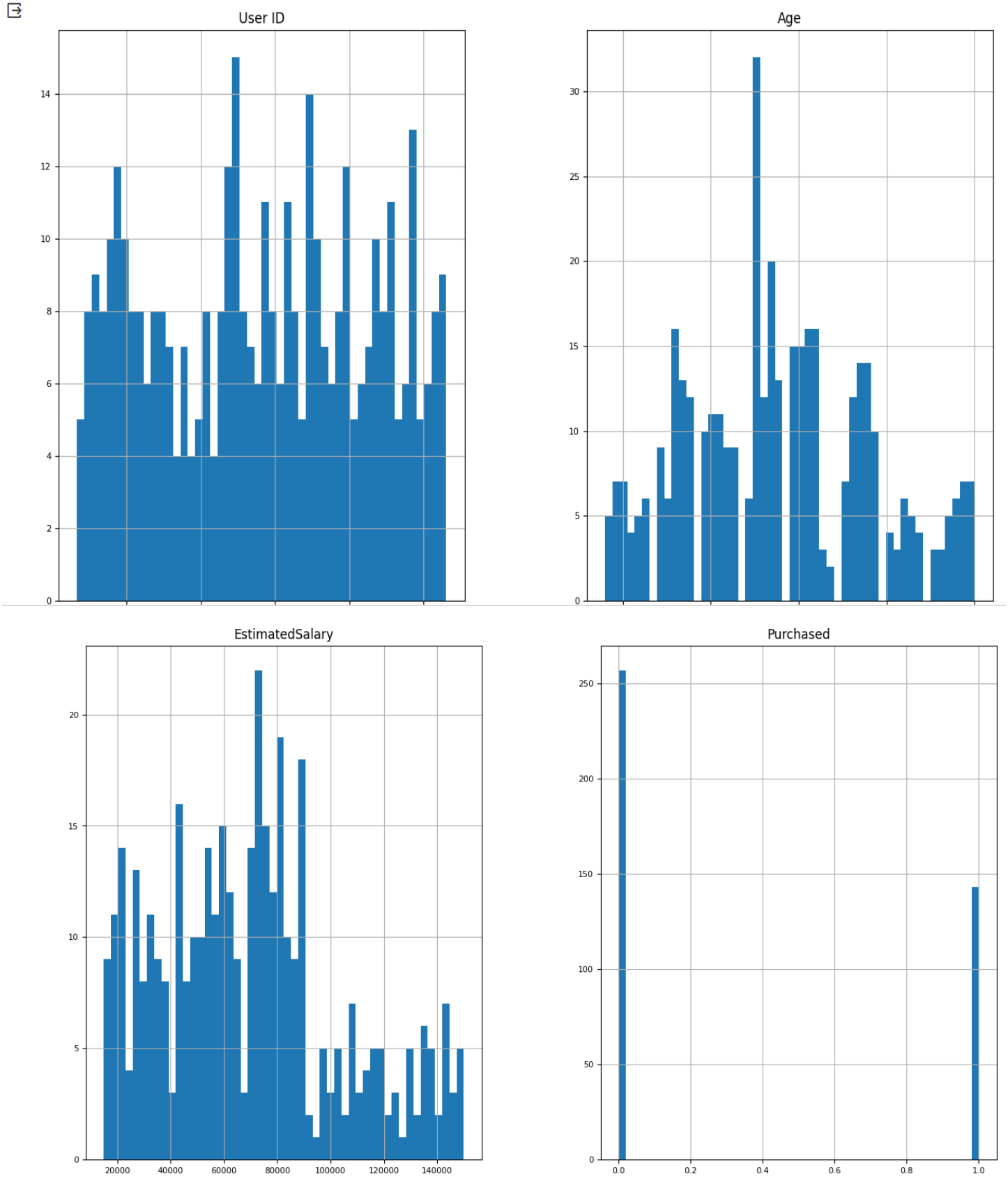


```
sns.distplot(df["Age"])
```

```
<ipython-input-11-cf0334540b62>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Age"])
<Axes: xlabel='Age', ylabel='Density'>
```



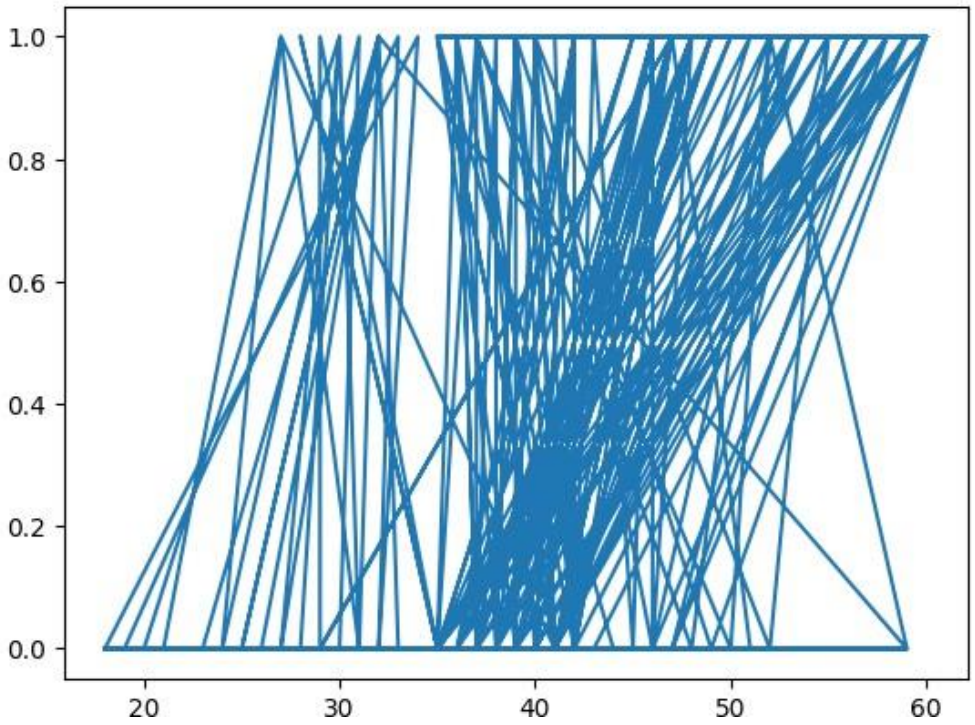
```
df.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8); # ; avoid having the matplotlib verbose informations
```



Bivariate analysis

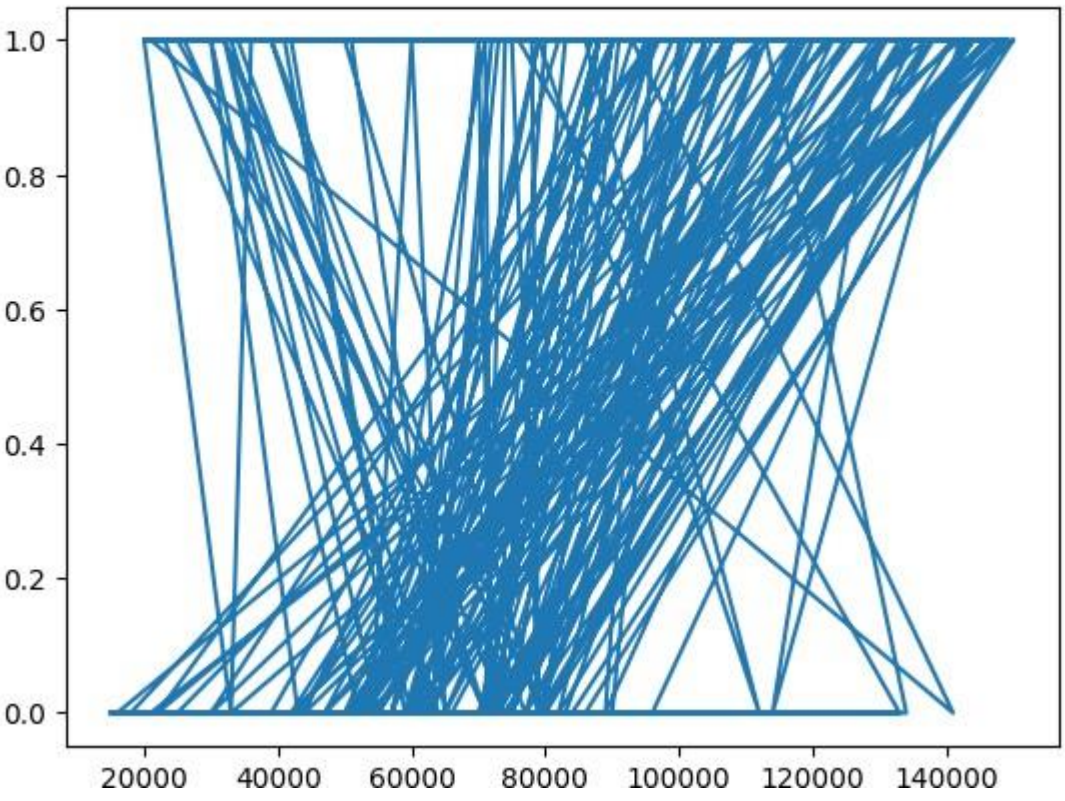
```
plt.plot(df.Age,df.Purchased)
```

```
[<matplotlib.lines.Line2D at 0x7b47c934a710>]
```



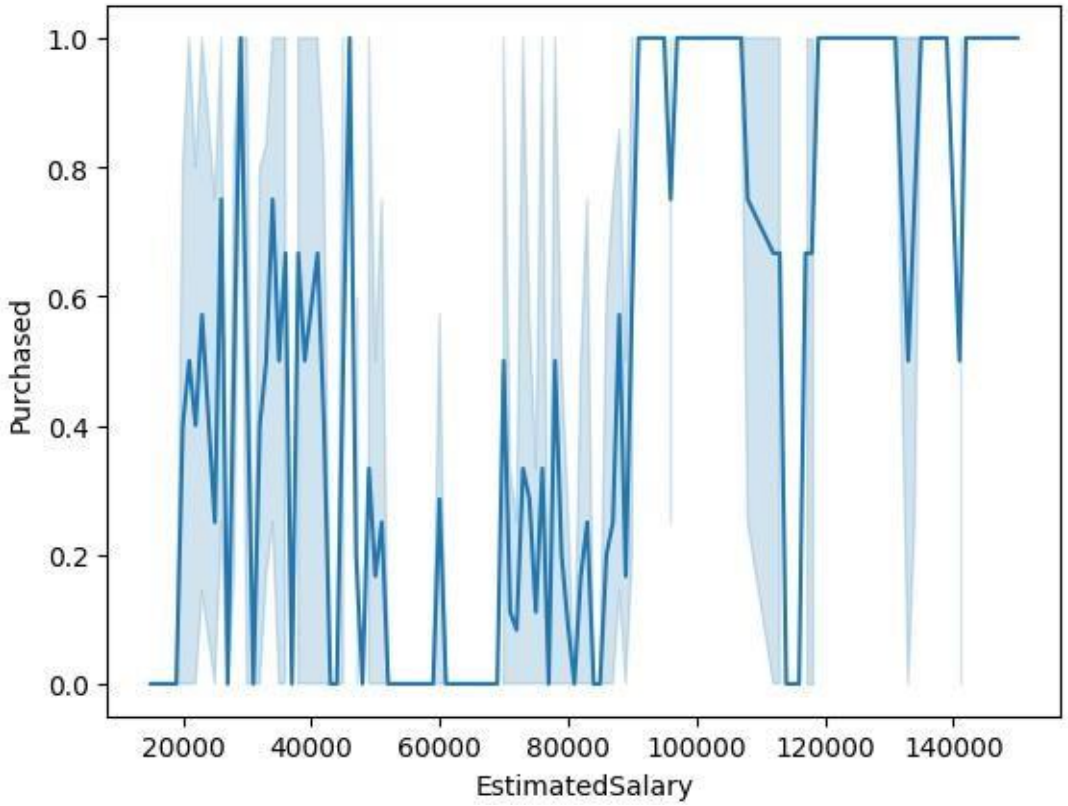

```
plt.plot(df.EstimatedSalary,df.Purchased)
```

```
[<matplotlib.lines.Line2D at 0x7b47c91afe20>]
```



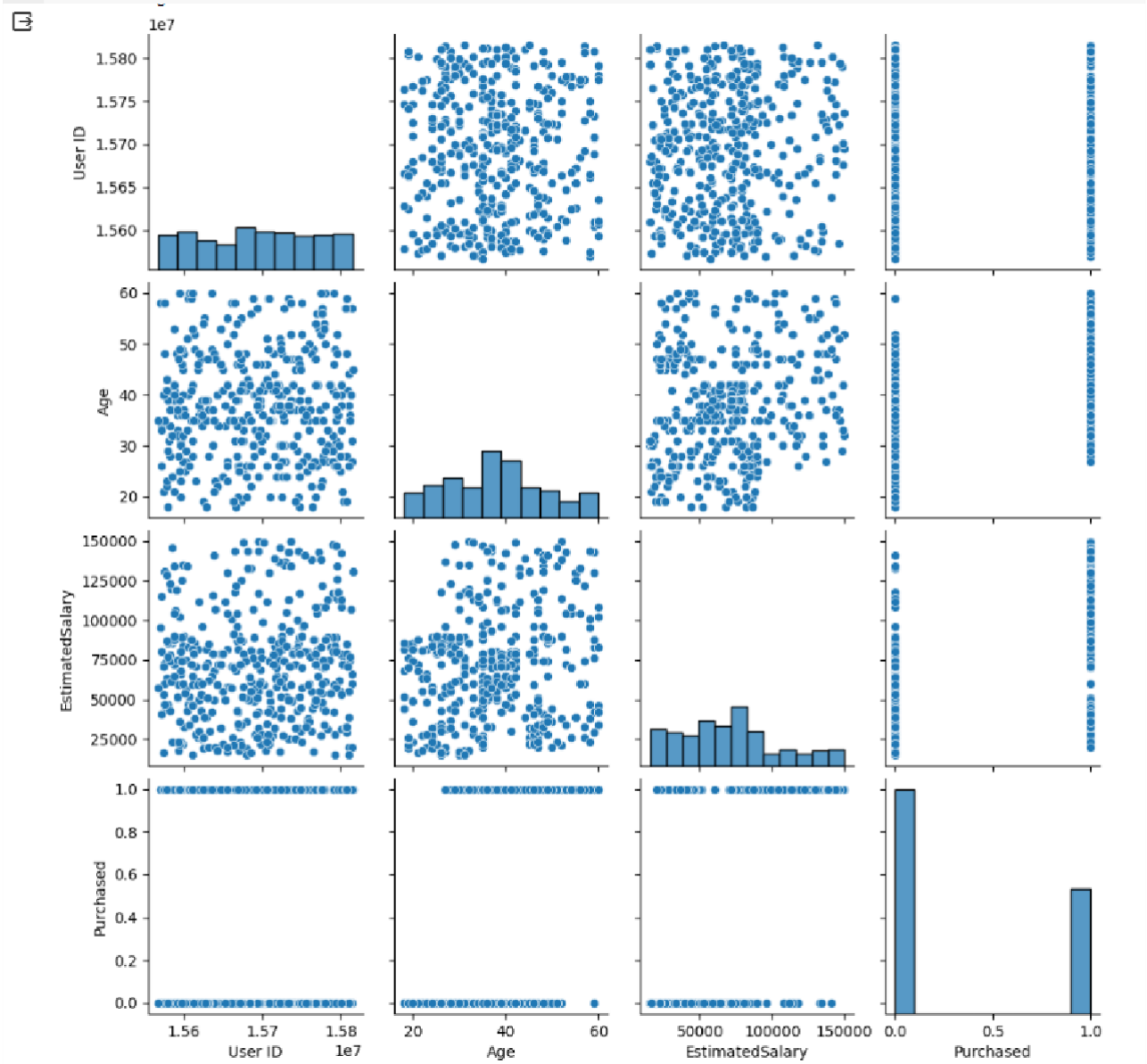
```
sns.lineplot(x="EstimatedSalary",y="Purchased",data=df)
```

```
<Axes: xlabel='EstimatedSalary', ylabel='Purchased'>
```



MULTIVARIATE ANALYSIS

```
sns.pairplot(df)
```

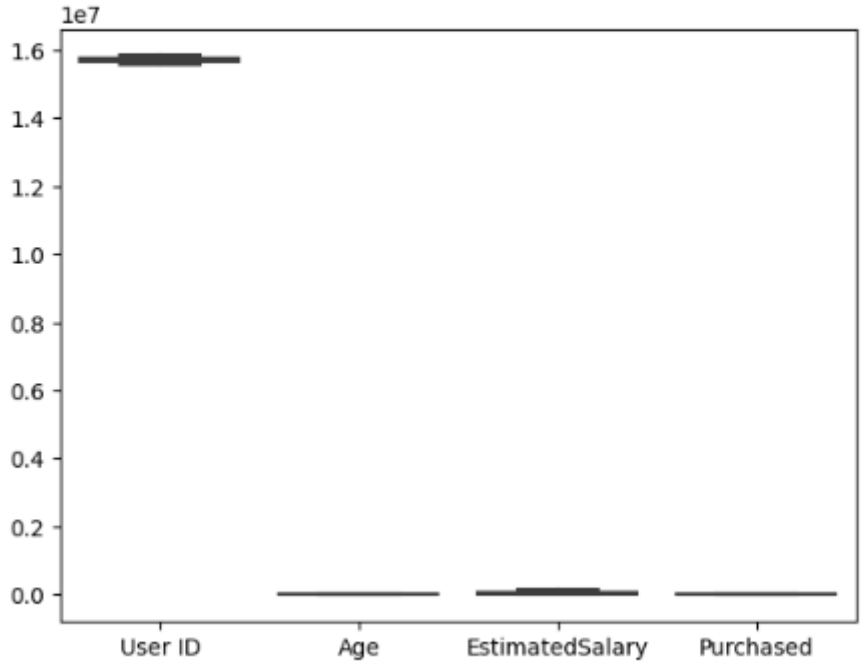


Activity 5: Outlier detection

OUTLIER DETECTION

```
sns.boxplot(df)
```

```
<Axes: >
```



From the figure we come to know that there are no outliers.Hence,they need not be treated

Activity 6: Splitting Dependent and independent features

Splitting the Dataset into dependent and independent variables

```
x=df.iloc[:,1:4]
x.head()
```

	Gender	Age	EstimatedSalary
0	Male	19	19000
1	Male	35	20000
2	Female	26	43000
3	Female	27	57000
4	Male	19	76000

```
y=df.Purchased
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Purchased, dtype: int64
```

Activity 7: Encoding categorical features

ENCODING THE CATEGORICAL DATA USING LABELENCODER()

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
x["Gender"]=le.fit_transform(x["Gender"])
x.head()
```

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000

Thus the gender feature has been encoded

Activity 8: Splitting data into Train and Test

SPLITTING THE DATASET INTO TRAINING SET AND TESTING SET

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=0)
```

```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(320, 3) (80, 3) (320,) (80,)
```

MODEL BUILDING

Our objective is to try working using various ml algorithms and figuring out the best algorithmsuited for this particular use case

1)Logistic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
    lr=LogisticRegression()
```

```
lr.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
[ ] pred=lr.predict(x_test)
    pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

```
[ ] #random value prediction
    lr.predict(ms.transform([[1,19,19000]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
array([0])
```

EVALUATION OF MODEL

```
[ ] from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
[ ] score1=accuracy_score(y_test,pred)
    score1
```

```
0.925
```

```
[ ] confusion_matrix(y_test,pred)
```

```
array([[58,  0],
       [ 6, 16]])
```

```
pd.crosstab(y_test,pred)
```

col_0	0	1
Purchased		
0	58	0
1	6	16

```
[ ] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        58
           1       1.00      0.73      0.84        22

    accuracy                           0.93        80
   macro avg       0.95      0.86      0.90        80
weighted avg       0.93      0.93      0.92        80
```

2) K_nearest neighbors[KNN]

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
y_pred=knn.predict(x_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

```
#random val prediction
knn.predict([[1,19,19000]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
array([1])
```

EVALUATNG THE MODEL

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] score=accuracy_score(y_pred,y_test)
    score
```

```
0.95
```

```
[ ] print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80
```

```
pd.crosstab(y_test,y_pred)
```

col_0	0	1
Purchased		
0	55	3
1	1	21

3) Decision tree classification

```
[ ] from sklearn.tree import DecisionTreeClassifier
    dtc=DecisionTreeClassifier()
    dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
pred2=dtc.predict(x_test)
pred2
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1])
```

```
[ ] #random value prediction
    dtc.predict(ms.transform([[1,19,19000]]))
```

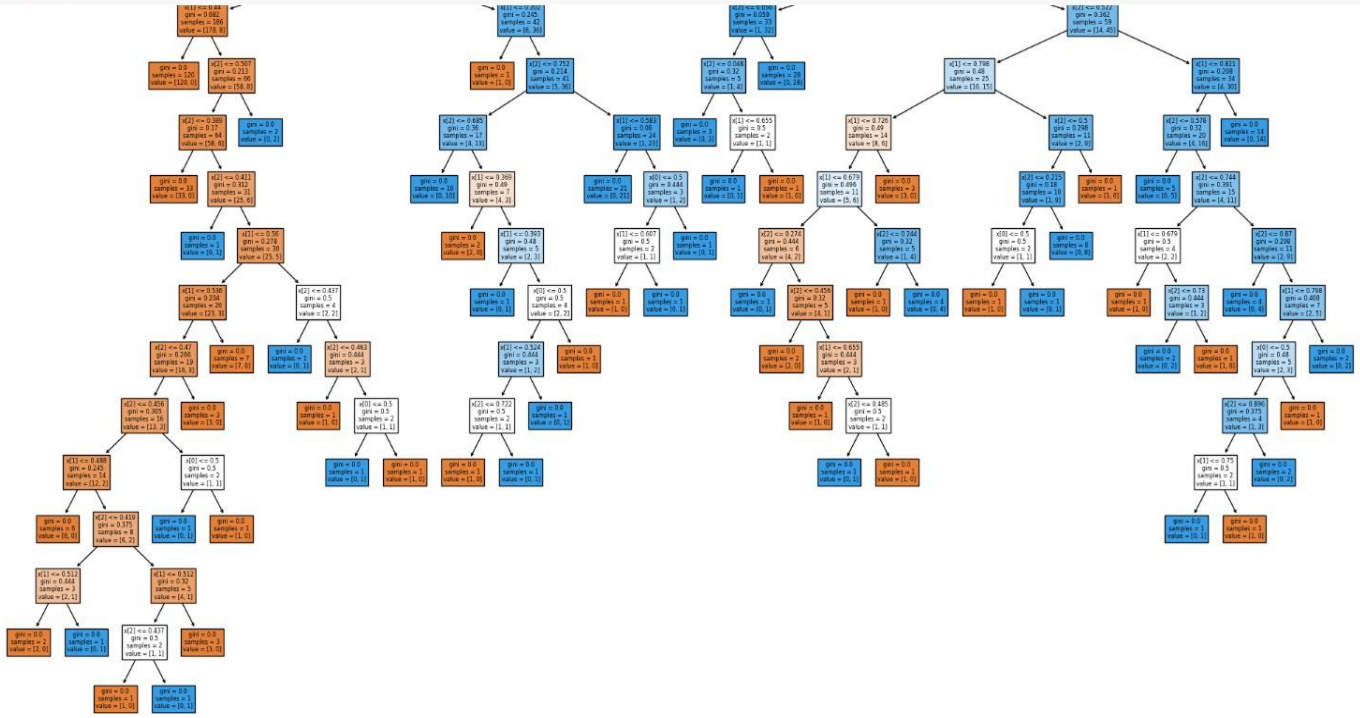
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
array([0])
```

```
[ ] score2=accuracy_score(y_test,pred2)
    score2
```

```
0.925
```

HYPER PARAMETER TUNING

```
[ ] from sklearn import tree
    plt.figure(figsize=(25,15))
    tree.plot_tree(dtc,filled=1)
```

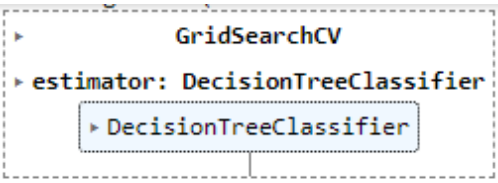


```
[ ] from sklearn.model_selection import GridSearchCV
    parameter={
        'criterion':['gini','entropy'],'splitter':['best','random'],'max_depth':[1,2,3,4,5],'max_features':['auto','sqrt','log2']

    }
```

```
[ ] grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring='accuracy')
```

```
[ ] grid_search.fit(x_train,y_train)
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
```




```
[ ] grid_search.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 4,
 'max_features': 'log2',
 'splitter': 'best'}
```

```
[ ] dtc_cv=DecisionTreeClassifier(criterion= 'gini',
     max_depth= 4,
     max_features='log2',
     splitter='best')
    dtc_cv.fit(x_train,y_train)
```

```
DecisionTreeClassifier(max_depth=4, max_features='log2')
```

```
pred_cv=dtc_cv.predict(x_test)
```

```
[ ] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        58
           1       1.00      0.73      0.84        22

    accuracy                           0.93        80
   macro avg       0.95      0.86      0.90        80
weighted avg       0.93      0.93      0.92        80
```

```
[ ] score_cv=accuracy_score(y_test,pred_cv)
    score_cv
```

```
0.95
```

4) random forest classification

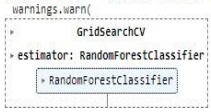
```
[ ] from sklearn.ensemble import RandomForestClassifier
    rfc=RandomForestClassifier()
```

```
[ ] forest_params=[{'max_depth':list(range(10,15)),'max_features':list(range(0,14))}]
```

```
[ ] rfc_cv=GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring='accuracy')
```

```
[ ] rfc_cv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: FitFailedWarning:
50 fits failed out of a total of 700.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
50 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py", line 340, in fit
    self._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'auto' (deprecated), 'sqrt'} or None. Got 0 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non-finite: [     nan 0.884375 0.8875   0.884375 0.890625 0.878125 0.88125  0.8875
 0.875    0.878125 0.884375 0.875    0.86875  0.878125      nan 0.884375
 0.890625 0.878125 0.878125 0.884375 0.878125 0.878125 0.88125  0.86875
 0.884375 0.884375 0.88125  0.88125       nan 0.884375 0.8875   0.878125
 0.878125 0.878125 0.875    0.878125 0.8875   0.878125 0.8625   0.865625
 0.884375 0.884375      nan 0.8875   0.8875   0.884375 0.865625 0.871875
 0.878125 0.878125 0.875    0.88125  0.884375 0.875    0.890625 0.8875
      nan 0.884375 0.8875   0.884375 0.884375 0.884375 0.878125 0.884375
 0.88125  0.875    0.88125  0.88125  0.88125  0.88125 ]
  warnings.warn(
```




```
[ ] pred3=rfc_cv.predict(x_test)
```

```
[ ] # print(classification_report(y_test,pred))
```

```
score4=accuracy_score(y_test,pred3)
score4
```

```
0.925
```

5) Support Vector Machine (SVM) classification

```
[ ] from sklearn.svm import SVC
    model=SVC(probability=True)
```

```
[ ] rand_list={"C":[2,3,4,5,7,8,10],"gamma":[0.1,0.2,0.3,0.5,0.6,0.8]}#c  values tells us about how the hyperplane is created
    #if c is less then low hyperplane is created.if c is high error is high
    #gamma value tells us how loosly the training data is fitted into the model
```

```
[ ] from sklearn.model_selection import RandomizedSearchCV
    rand_search=RandomizedSearchCV(model,param_distributions=rand_list,n_iter=20,cv=5)
```

```
    rand_search.fit(x_train,y_train)
```

```
RandomizedSearchCV
  estimator: SVC
    SVC
```

```
[ ] pred5=rand_search.predict(x_test)
    pred5
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

```
[ ] rand_search.predict(ms.transform([[1,19,19000]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
array([0])
```

```
[ ] score5=accuracy_score(y_test,pred5)
    score5
```

```
0.95
```

6) Naive bayes

```
[ ] from sklearn.naive_bayes import GaussianNB
    nb=GaussianNB()
    nb.fit(x_train,y_train)
```

```
GaussianNB()
```

```
[ ] pred6=nb.predict(x_test)
    pred6
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

```
    #random value prediction
    nb.predict(ms.transform([[1,19,19000]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
array([0])
```

```
[ ] cm6 = confusion_matrix(y_test, pred6)
```

```
[ ] print(cm6)
```

```
[[56  2]
 [ 4 18]]
```

```
[ ] score7=accuracy_score(y_test,pred6)
    score7
```

```
0.925
```

```
[ ] score7=accuracy_score(y_test,pred6)
    score7
```

```
0.925
```

```
[ ] pd.crosstab(y_test,pred)
```

col_0	0	1
Purchased		
0	58	0
1	6	16

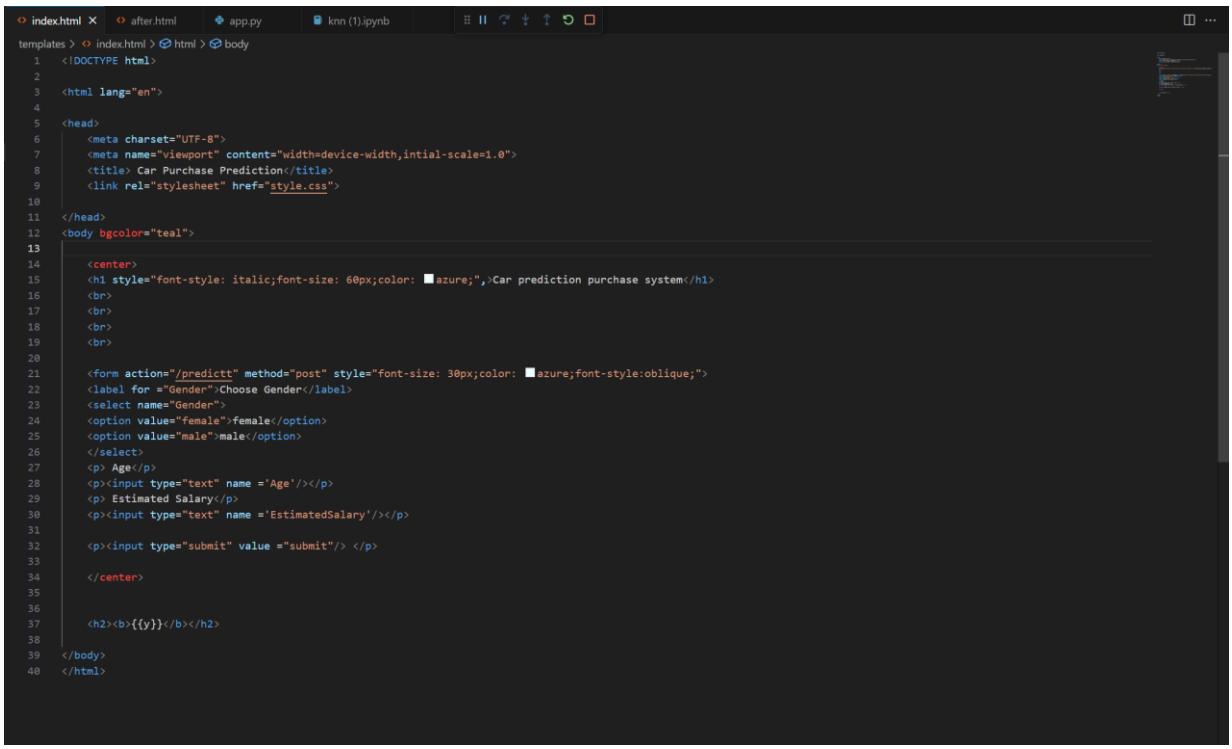
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        58
           1       1.00      0.73      0.84        22

    accuracy                           0.93        80
   macro avg       0.95      0.86      0.90        80
weighted avg       0.93      0.93      0.92        80
```

Inference

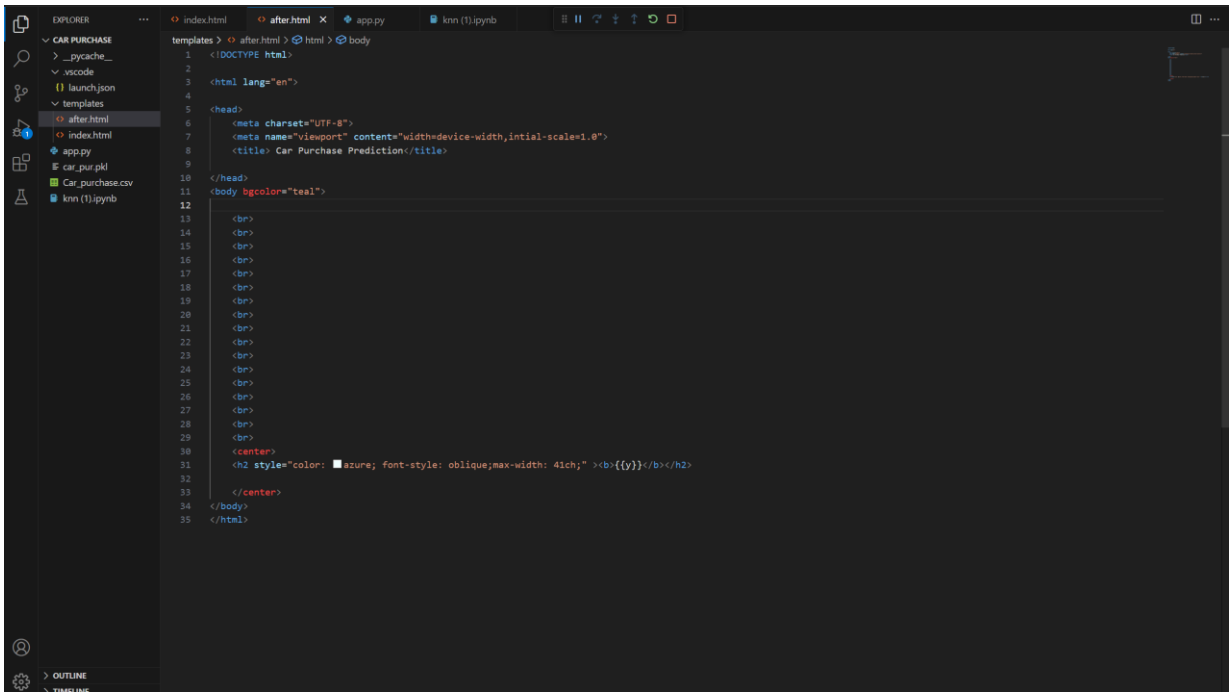
Among all implemented algorithms , Support Vector Machine(SVC) and KNN perform well on Car Purchase Dataset . Accuracy of KNN and SVM : 95%

HTML code
homepage html template



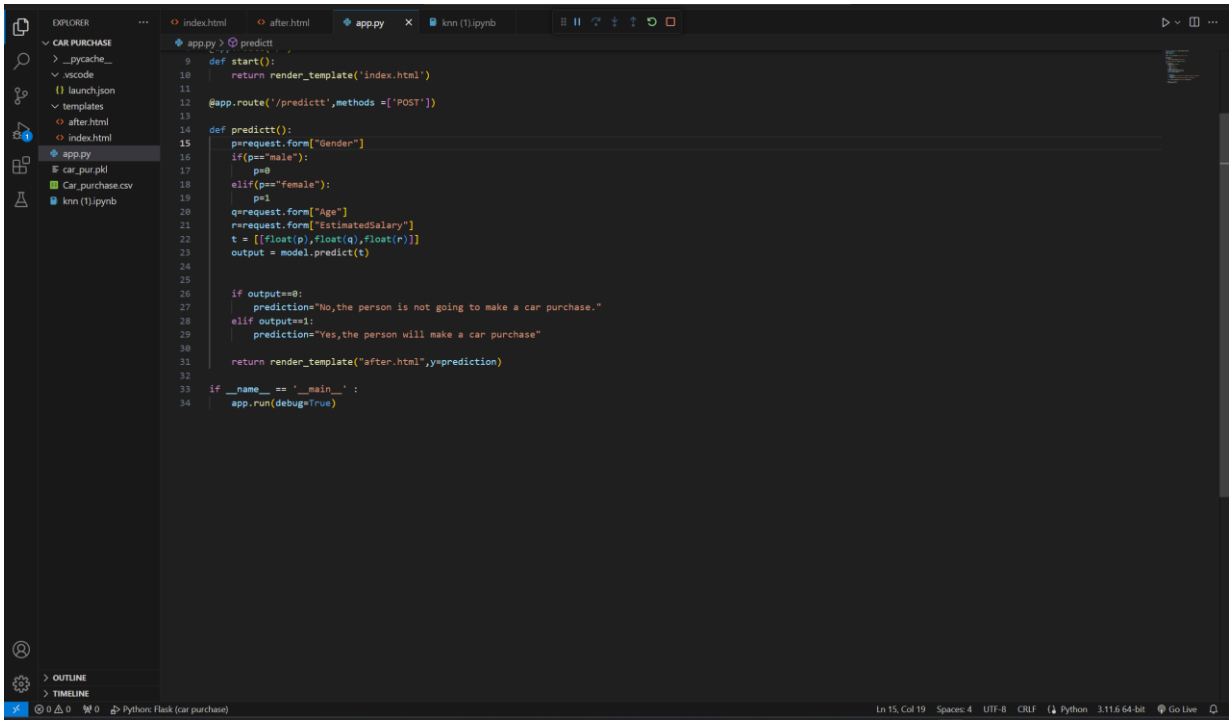
```
<!DOCTYPE html>

<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,intial-scale=1.0">
    <title> Car Purchase Prediction</title>
    <link rel="stylesheet" href="style.css">

</head>
<body bgcolor="teal">

    <center>
    <h1 style="font-style: italic;font-size: 60px;color: azure;",>Car prediction purchase system</h1>
    <br>
    <br>
    <br>
    <br>

    <form action="/predictt" method="post" style="font-size: 30px;color: azure;font-style:oblique;">
    <label for ="Gender">Choose Gender</label>
    <select name="Gender">
    <option value="female">female</option>
    <option value="male">male</option>
    </select>
    <p> Age</p>
    <p><input type="text" name ='Age'/></p>
    <p> Estimated Salary</p>
    <p><input type="text" name ='EstimatedSalary'/></p>

    <p><input type="submit" value ="submit"/> </p>

    </center>


    <h2><b>{{y}}</b></h2>

</body>
</html>
```

Result page



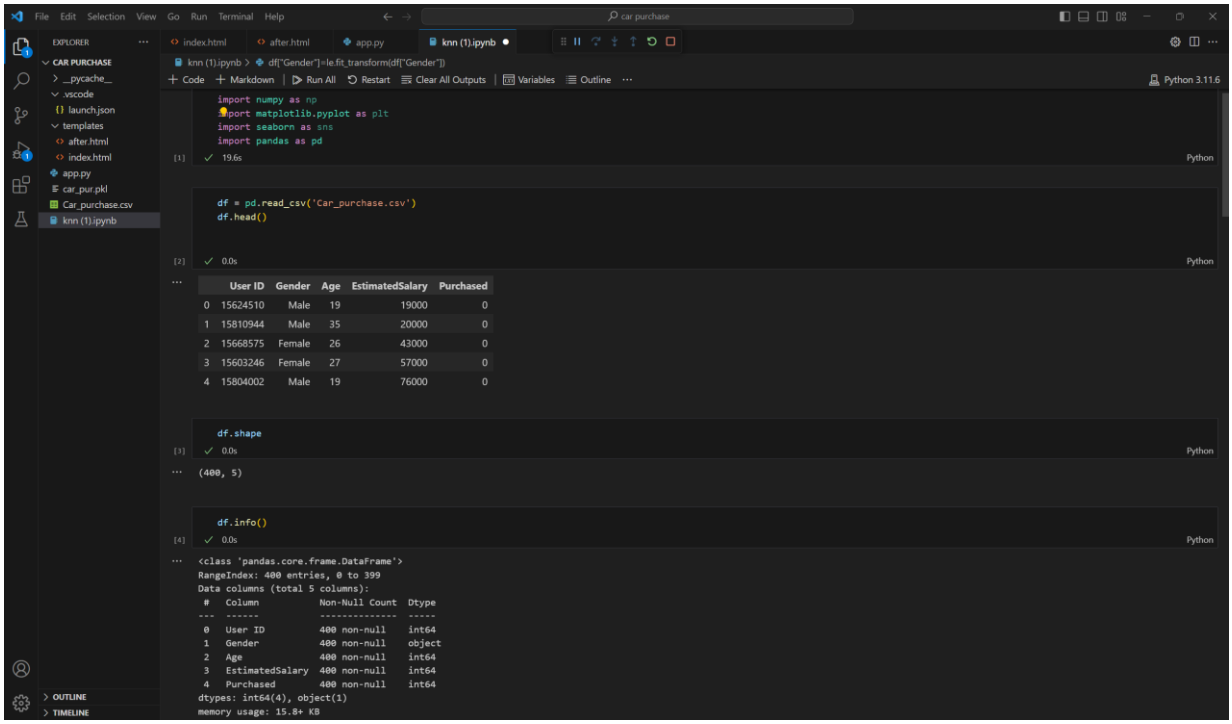
```
<!DOCTYPE html>

<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,intial-scale=1.0">
    <title> Car Purchase Prediction</title>

</head>
<body bgcolor="teal">

    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <center>
    <h2 style="color: azure; font-style: oblique;max-width: 41ch;" ><b>{{y}}</b></h2>

    </center>
</body>
</html>
```

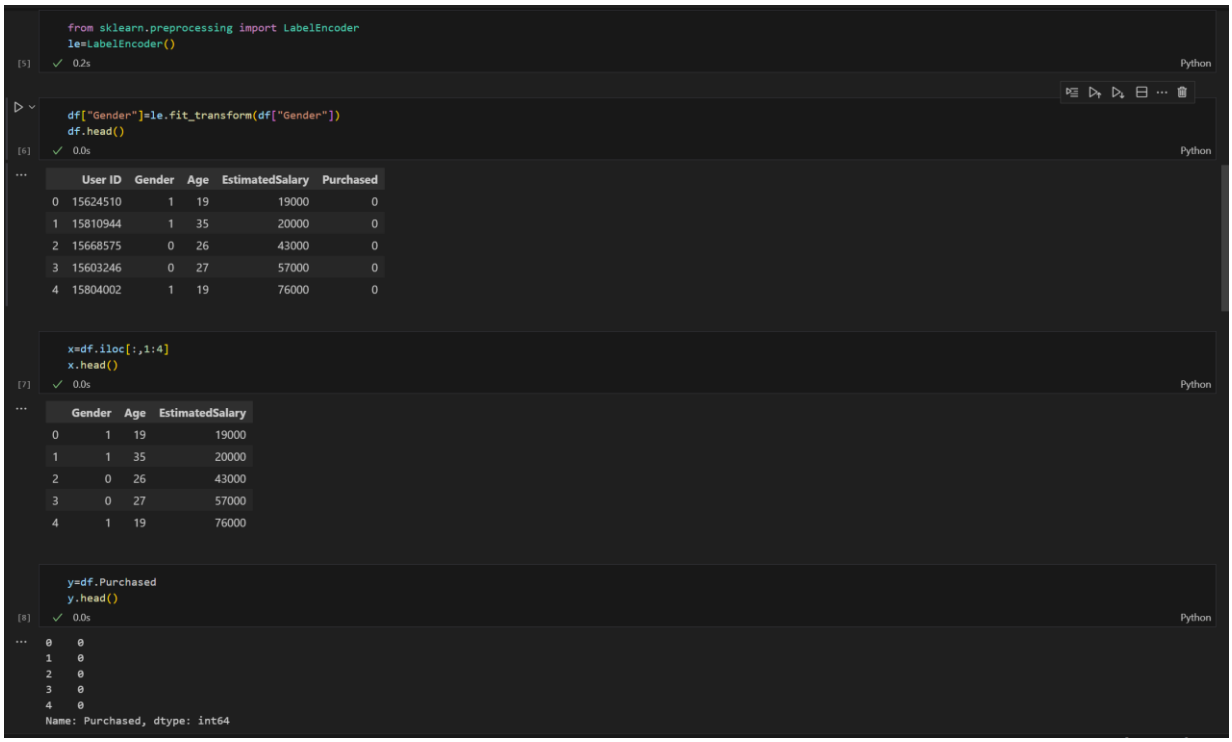
Python code(flask)

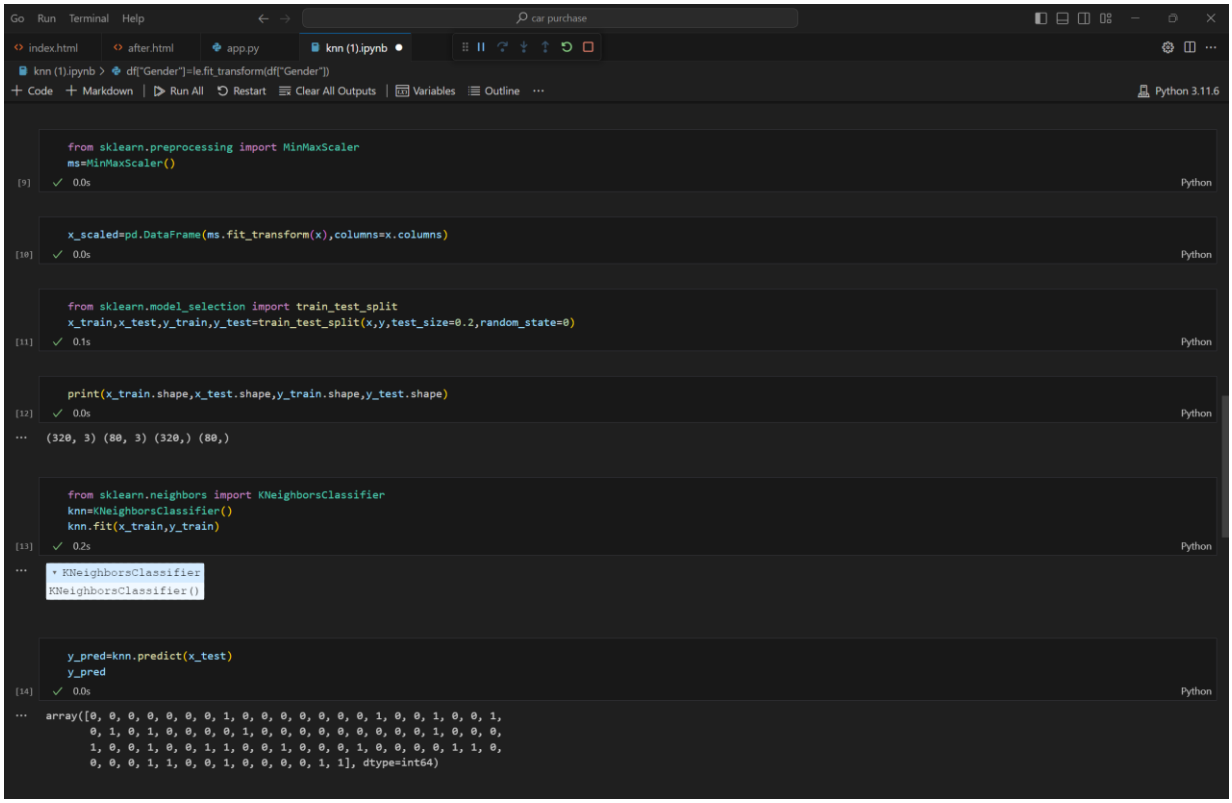


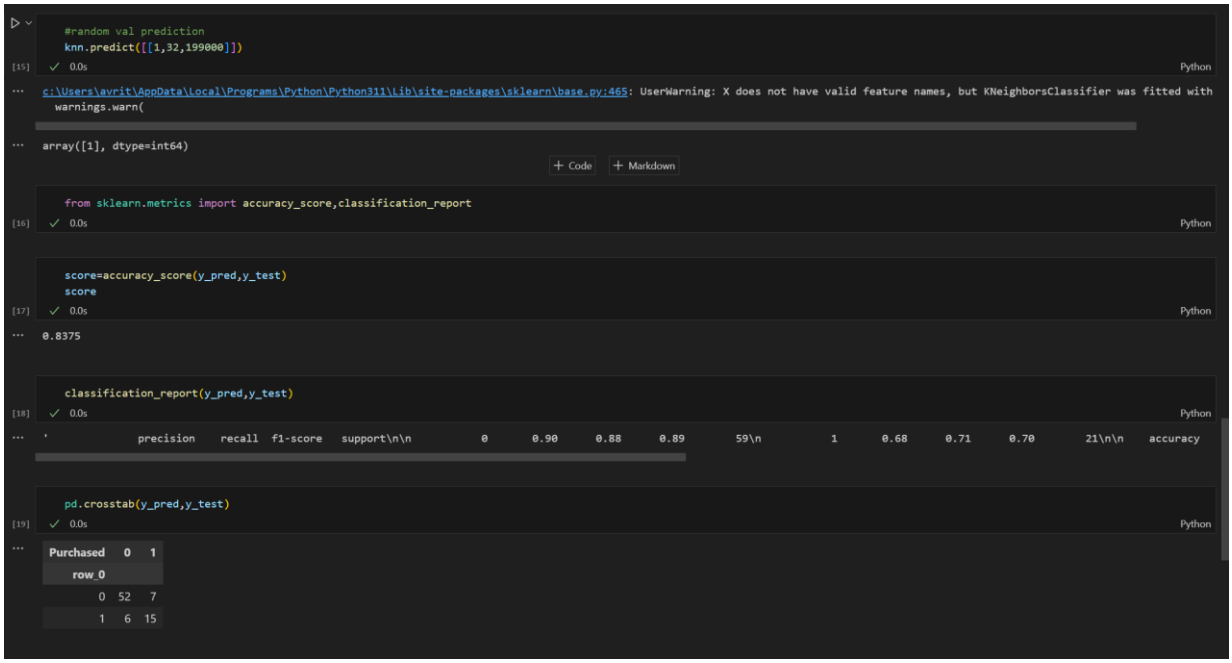
```
def start():
    return render_template('index.html')

@app.route('/predictt',methods =['POST'])

def predictt():
    p=request.form["Gender"]
    if(p=="male"):
        p=0
    elif(p=="female"):
        p=1
    q=request.form["Age"]
    r=request.form["EstimatedSalary"]
    t = [[float(p),float(q),float(r)]]
    output = model.predict(t)


    if output==0:
        prediction="No,the person is not going to make a car purchase."
    elif output==1:
        prediction="Yes,the person will make a car purchase"

    return render_template("after.html",y=prediction)

if __name__ == '__main__' :
    app.run(debug=True)
```

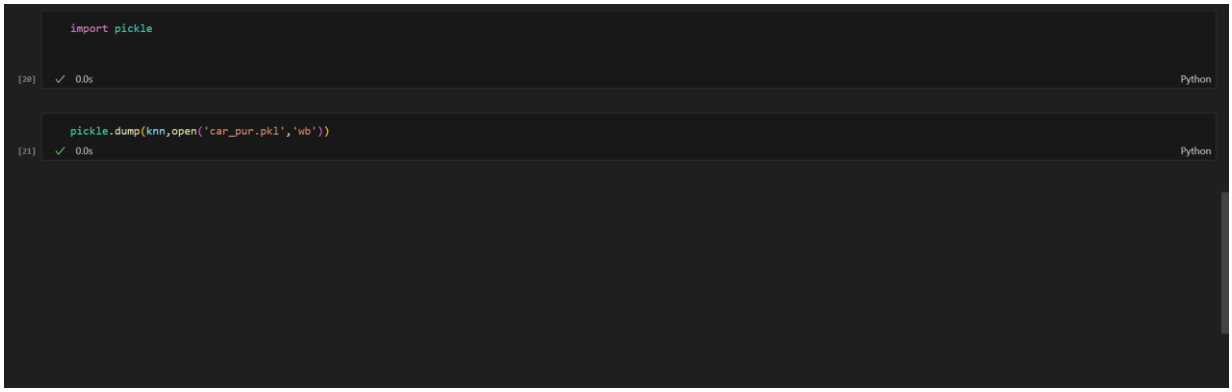
Knn jupyter source file


File Edit Selection View Go Run Terminal Help
car purchase
EXPLORER
CAR PURCHASE
__pycache__
.vscode
launch.json
templates
after.html
index.html
app.py
car_pur.pkl
Car_purchase.csv
knn (1).ipynb
index.html
after.html
app.py
knn (1).ipynb
knn (1).ipynb > df["Gender"]=le.fit_transform(df["Gender"])
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python 3.11.6
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
[1] 19.6s
Python
df = pd.read_csv('Car_purchase.csv')
df.head()
[2] 0.0s
Python
User ID Gender Age EstimatedSalary Purchased
0 15624510 Male 19 19000 0
1 15810944 Male 35 20000 0
2 15668575 Female 26 43000 0
3 15603246 Female 27 57000 0
4 15804002 Male 19 76000 0
df.shape
[3] 0.0s
Python
(400, 5)
df.info()
[4] 0.0s
Python
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
Column Non-Null Count Dtype
0 User ID 400 non-null int64
1 Gender 400 non-null object
2 Age 400 non-null int64
3 EstimatedSalary 400 non-null int64
4 Purchased 400 non-null int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
OUTLINE
TIMELINE


from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
[5] 0.2s
Python
df["Gender"]=le.fit_transform(df["Gender"])
df.head()
[6] 0.0s
Python
User ID Gender Age EstimatedSalary Purchased
0 15624510 1 19 19000 0
1 15810944 1 35 20000 0
2 15668575 0 26 43000 0
3 15603246 0 27 57000 0
4 15804002 1 19 76000 0
x=df.iloc[:,1:4]
x.head()
[7] 0.0s
Python
Gender Age EstimatedSalary
0 1 19 19000
1 1 35 20000
2 0 26 43000
3 0 27 57000
4 1 19 76000
y=df.Purchased
y.head()
[8] 0.0s
Python
0 0
1 0
2 0
3 0
4 0
Name: Purchased, dtype: int64


Go Run Terminal Help
car purchase
index.html
after.html
app.py
knn (1).ipynb
knn (1).ipynb > df["Gender"]=le.fit_transform(df["Gender"])
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
Python 3.11.6
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
[9] 0.0s Python
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
[10] 0.0s Python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
[11] 0.1s Python
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
[12] 0.0s Python
(320, 3) (80, 3) (320,) (80,)
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
[13] 0.2s Python
KNeighborsClassifier
KNeighborsClassifier()
y_pred=knn.predict(x_test)
y_pred
[14] 0.0s Python
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1], dtype=int64)


#random val prediction
knn.predict([[1,32,199000]])
[15] 0.0s Python
c:\Users\avrit\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with
warnings.warn(
array([1], dtype=int64)
+ Code + Markdown
from sklearn.metrics import accuracy_score,classification_report
[16] 0.0s Python
score=accuracy_score(y_pred,y_test)
score
[17] 0.0s Python
0.8375
classification_report(y_pred,y_test)
[18] 0.0s Python
' precision recall f1-score support\n\n 0 0.90 0.88 0.89 59\n 1 0.68 0.71 0.70 21\n\n accuracy
pd.crosstab(y_pred,y_test)
[19] 0.0s Python
Purchased 0 1
row_0
0 52 7
1 6 15



GitHub link - https://github.com/smartinternz02/SI-GuidedProject-590861-1697117192

Project demo link - https://drive.google.com/file/d/1EePenM2WsV62pucaZ0oZHB3UErRv7oQl/view?usp=sharing